In this project, I built an intelligent agent that uses reinforcement learning to solve a maze. The goal was for a pirate character to find a treasure placed in the bottom-right corner of the maze without running into blocked paths. The maze was created using a matrix where 1.0 meant a free space and 0.0 meant an obstacle. I trained the pirate agent using a Deep Q-Learning algorithm, which allowed it to learn the best moves over time. At first, the pirate made random choices, but as it learned from rewards and mistakes, it began to find better paths to the treasure.

The code worked well and ran without errors. I used good programming practices by organizing everything into separate classes. The TreasureMaze class controlled the maze and the pirate's position. The GameExperience class stored past episodes and helped the agent learn from its experience. I also wrote a qtrain() function that managed the training process. I made sure to add clear comments throughout the code to explain what each part does. This helped with understanding the logic and made the program easier to update or fix if needed. Following these best practices made the project feel more like something that could be used in a real-world setting.

When comparing the agent's learning process to how a human might solve a maze, there are some big differences. Humans often use memory, trial and error, or logic to figure out a path. The pirate agent had to start from scratch and learn only from rewards and penalties. According to Botvinick et al. (2019), reinforcement learning in machines is a lot like how our brains work. We have one system that reacts quickly and another that takes more time to plan. The pirate agent was more like the second system. It learned from experience and tried to choose actions that led to better outcomes. While the agent doesn't think like a person, it still learns to make smarter choices over time.

The main reason for creating the agent was to show how deep learning can help solve problems where there isn't a clear solution at the start. The agent learned by playing the game over and over, each time improving a little. It used a technique called experience replay to remember what happened and learn from it. The training results showed that the more it practiced, the better it got. The win rate increased and the loss went down. This shows that the algorithm worked well. As Botvinick et al. (2019) explain, this type of learning is powerful because it helps both humans and machines improve performance through feedback.

Overall, this project showed that reinforcement learning is a strong way to train intelligent systems. By practicing and learning from feedback, the pirate agent went from making random choices to reaching the treasure almost every time. This type of learning can be useful for many real-world problems, like robots learning to move or apps that suggest good decisions. The research from Botvinick and his team helped me understand how this learning process connects to how people learn too. It made the project more meaningful and showed the value of using reinforcement learning in smart systems.

Reference

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019). Reinforcement learning, fast and slow. *Trends in Cognitive Sciences, 23*(5), 408–422. https://doi.org/10.1016/j.tics.2019.02.006